ELSEVIER

Theoretical Computer Science 255 (2001) 107–123

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Branch and bound on the network model

Sanjay Jain

*School of Computing, National University of Singapore, Lower Kent Ridge Road, Singapore 119260, Singapore*

Received October 1997; revised September 1998
Communicated G. Ausiello

**Abstract**

Karp and Zhang developed a general randomized parallel algorithm for solving branch and bound problems. They showed that with high probability their algorithm attained optimal speedup within a constant factor (for $p \leqslant n/(\log n)^c$, where $p$ is the number of processors, $n$ is the "size" of the problem, and $c$ is a constant). Ranade later simplified the analysis and obtained a better processor bound. Karp and Zhang's algorithm works on models of computation where communication cost is constant. The present paper considers the Branch and Bound problem on networks where the communication cost is high. Suppose sending a message in a $p$ processor network takes $G = O(\log p)$ time and node expansion (defined below) takes unit time (other operations being free). Then a simple randomized algorithm is presented which is, asymptotically, nearly optimal for $p = O(2^{\log^c n})$, where $c$ is any constant $< 1/3$ and $n$ is the number of nodes in the input tree with cost no greater than the cost of the optimal leaf in the tree. 

*Keywords:* Branch and bound; Parallel algorithms

## 1. Introduction

Branch and Bound algorithms are frequently used for solving optimization problems. Because of the importance of several such optimization problems, and the fact that most of these jobs are computation intensive, it is useful to look at parallelizing such algorithms.

Karp and Zhang [4] presented a simple randomized optimal algorithm for such problems. They showed that the algorithm was optimally fast (upto a constant multiplicative factor) on every problem instance with high probability. Their analysis was later sim-

*E-mail address:* sanjay@comp.nus.edu.sg (S. Jain).

0304-3975/01/$ - see front matter 
PII: S0304-3975(99)00158-9

plified and improved by Ranade [6]. (Karp and Zhang also improved their original analysis in the journal version of their paper [5].)

Karp and Zhang (Ranade) assumed a constant time communication between the processors (with local memory only). However, in reality networks are not fully connected and communication cost could go up depending on the number of processors. We consider Branch and Bound on a model in which communication cost depends on the number of processors. In Section 2 we present the generic Branch and Bound problem and our model. Section 3 gives a lemma useful in proving our result. Our algorithm is presented in Section 4. Section 5 gives the analysis of the algorithm.

## 2. Model

Karp and Zhang model a generic Branch and Bound problem as follows. A tree $H$ with costs associated with each node is given. The cost function has the property that if $v$ is the parent of $w$ then $\mathrm{Cost}(v) < \mathrm{Cost}(w)$. The goal is to find a least cost leaf. For simplicity we assume that all costs are different. The input to the algorithm is the root of $H$, and the other nodes are generated during execution using a procedure called node expansion. When this procedure is applied to a node $v$, it either determines that $v$ is a leaf, or it generates the children of $v$ along with the associated costs. A node can be expanded only if it is a root or it has been generated by expanding its parent earlier. We assume that the degree of nodes of $H$ are bounded by a constant. Specifically, we assume that $H$ is a binary tree, even though our analysis works for any $b$-ary tree. In many applications (such as in branch-and-bound algorithm for Travelling Salesman Problem) the degree of the branch and bound tree can be made constant by replacing a node with $d$ children by a binary tree with height $\log d$ (this increases the height by a multiplicative factor logarithmic in the maximum degree). Let $\tilde{H}$ denote the subtree of $H$ formed using nodes with cost no greater than the cost of the minimal cost leaf. Let $n$ denote the number of nodes in $\tilde{H}$ and $h$ denote the height of $\tilde{H}$. It is easy to see that any sequential algorithm for branch and bound must do at least $n$ node expansions, and that any parallel algorithm must take at least $\Omega(n/p + h)$ steps, where $p$ is the number of processors used. In a parallel computation model with communication cost being constant, Karp and Zhang [4] gave an algorithm which achieves the above lower bound for $p < n/(\log n)^k$, for some constant $k$. Ranade [6] later simplified the analysis and improved the processor bound to $n/\log n$. Karp and Zhang [5] also improved their original analysis to get a processor bound of $n/\log n$.

We use the following model of parallel computation (see [1–3] for similar models). There are $p$ processors numbered from 1 to $p$. Each processor has local memory and there is no global memory. There is no global control but the processors operate synchronously (this is only for the analysis, and not for the algorithm). We assume that the processors have capability to do independent random coin tosses. Expanding a node takes one unit of time, sending a message takes $G$ units of time, and receiving a

message is free.[1] Thus, a message sent by processor $P_i$ to processor $P_j$ at time $t$ will be received by processor $P_j$ at time $t+G$ and processor $P_i$ is unavailable for doing any work between time $t$ and $t+G$. In any message, a processor can transmit at most one node of the tree $H$.

In our model, as in the model of Karp and Zhang and that of Ranade, we charge a processor just for node expansions (1 unit of time) and communication. Everything else is free. This is based on the assumption that in real applications cost of communication and node expansion would dominate.

Karp and Zhang's algorithm (as modified by Ranade) is essentially the algorithm given below for the case when $G=1$. Our main result is that for communication time logarithmic in the number of processors, branch and bound can be implemented in nearly optimal randomized time, for processor bound of $O(2^{(\log n)^c})$ (for $c<1/3$).

**Notation:** All logarithms in this paper are base 2. Size of a tree $T$, denoted Size($T$), is the number of nodes in $T$. For a tree or forest $F$ and node $v$ in $F$, let Sizesub($F,v$), denote the size of the subtree rooted at $v$ in $F$. Level of node $v$ in tree $T$ is defined as the distance of $v$ from the root of $T$. Thus level of root is 0. Height of a tree is the maximum over the level of the nodes of the tree. A node $v$ is an ancestor of node $u$ in tree $T$, iff $v$ lies in the unique path from root of $T$ to $v$. $v$ is a proper ancestor of $u$, iff $v$ is an ancestor of $u$ but $u \neq v$. A node $u$ is descendant of $v$, iff $v$ is ancestor of $u$. A node $u$ is proper descendant of $v$, iff $v$ is proper ancestor of $u$.

## 3. A useful lemma

In this section we prove a lemma which is useful in bounding the size of trees obtained in the analysis of our algorithm.

For a tree $T$, let $F_T^G$ denote a forest obtained when each edge of $T$ is retained with probability $1-1/G$. Let $P_T^G(w)$ denote the probability that the tree in $F_T^G$ containing the root of $T$ has size more than $w$.

In this section we will be proving a lemma (Lemma 8 below) which gives a reasonable upper bound on $P_T^G(w)$, for binary trees $T$ and certain values of $w$.

Before giving the lemma for general binary trees, we will first consider "balanced" trees (which may be non-binary).

**Definition 1.** A tree $T$ is said to be *r-balanced* iff for any node $v$ in the tree
(i) for all children $u$ of $v$, $\text{Sizesub}(T,u) \leqslant 2\lceil \text{Sizesub}(T,v)/r \rceil - 1$;
(ii) there are at most two children $u$ of $v$ with $\text{Sizesub}(T,u) < \lceil \text{Sizesub}(T,v)/r \rceil$.

[1] Anonymous referee pointed out that the model considered in this paper is known as Telephone model in the literature. Our algorithm and analysis can be easily modified to work for the case when receiving a message costs $G$ units of time, as long as simultaneous multiple receives are allowed or when received messages are queued in the input buffer according to a priority based on the cost of the nodes. To keep the presentation simple we do not consider such variants in this paper. At present we do not know what other models (for receives) our algorithm/analysis can be made to work.

Thus almost all children of any node in a balanced tree have nearly same size. For a $r$-balanced tree $T$, let us color any non-root node $u$ red, if $\text{Sizesub}(T,u) < \lceil \text{Sizesub}(T, \text{parent}(u))/r \rceil$. Other nodes (including the root) are colored blue.

The following proposition gives some properties of $r$-balanced trees.

**Proposition 2.** *Suppose $r \geqslant 5$. Suppose $T$ is an $r$-balanced tree. In the following, reference to children, parent and level of a node refers to tree $T$. Suppose $\text{Size}(T) = l$. Suppose $v$ is a node in $T$ and $\text{Sizesub}(T,v) = s$.*

(a) *Each node in $T$ has at most two children with red color, and at most $r$ children of blue color.*

(b) *For any red colored child $u$ of $v$, $\text{Sizesub}(T,u) \leqslant \lceil s/r \rceil - 1 < \frac{s}{r}$. For any blue colored child $u$ of $v$, $\lceil s/r \rceil \leqslant \text{Sizesub}(T,u) \leqslant 2\lceil s/r \rceil - 1 \leqslant 1 + (2s/r)$.*

(c) $\text{Sizesub}(T,u) \leqslant \text{Sizesub}(T,v)/2$, *for any child $u$ of $v$.*

(d) *The height of the tree $T$ is $\leqslant \log l$.*

(e) $\sum_{\{u \mid u \text{ is red colored child of } v\}} \text{Sizesub}(T,u) \leqslant 2s/r$.

(f) *The total number of nodes which have a red colored ancestor is at most $2l(\log l)/r$.*

**Proof.** (a), (b) follow from the definition of balanced tree and red/blue nodes.

(c) By part (b), for any child $u$ of $v$ in $T$, $\text{Sizesub}(T,u) \leqslant 2\lceil \text{Sizesub}(T,v)/r \rceil - 1 \leqslant \text{Sizesub}(T,v)/2$ (note that $\text{Sizesub}(T,v)$ must be at least 2, for $v$ to have a child in $T$).

(d) Follows from part (c).

(e) Follows using part (a) and (b).

(f) It follows from part (e) that, for each level $i$ in $T$, $\sum_{\{u \mid u \text{ is red colored and at level } i\}} \text{Sizesub}(T,u) \leqslant 2l/r$. Since by part (d), the height of the tree $T$ is at most $\log l$, part (f) follows. □.

**Lemma 3.** *Suppose $T$ is an $r$-balanced tree. Suppose $v$ is a node in $T$ and $\text{Sizesub}(T,v) = s$.*

(a) $\min\{r/3, s-1\} \leqslant$ *number of children of $v$ in $T \leqslant r+2$.*

(b) *Suppose $v$ has $k$ blue children in $T$. Then, for any blue child $u$ of $v$ in $T$, $\text{Sizesub}(T,u) \geqslant (\text{Sizesub}(T,v) - 1)/(2k+1)$.*

**Proof.** (a) The upper bound on the number of children follows from Proposition 2(a). Since the size of subtree rooted at any child of $v$ in $T$ is at most $2\lceil s/r \rceil - 1$, the degree of $v$ is at least $(s-1)/(2\lceil s/r \rceil - 1)$. For $r \geqslant s$, $(s-1)/(2\lceil s/r \rceil - 1) = s - 1$; and for $r < s$, $(s-1)/(2\lceil s/r \rceil - 1) \geqslant (s-1)/(2((s-1)/r + 1) - 1) \geqslant r/3$.

(b) Fix any blue child $u$ of $v$. Note that for any blue child $u'$ of $v$, $\text{Sizesub}(T,u') \leqslant 2\,\text{Sizesub}(T,u)$ (by Proposition 2(b)). Also, for any red child $u'$ of $v$, $\text{Sizesub}(T,u') \leqslant \text{Sizesub}(T,u)$. Let $S_b = \{u' \mid u' \text{ is a blue child of } v\}$, and $S_r = \{u' \mid u' \text{ is a red child of } v\}$. Now, $\text{Sizesub}(T,v) = 1 + \sum_{u' \in S_b - \{u\}} [\text{Sizesub}(T,u')] + \text{Sizesub}(T,u) + \sum_{u' \in S_r}$

$[\text{Sizesub}(T,u')] \leqslant 1+\sum_{u'\in S_b-\{u\}}[2\,\text{Sizesub}(T,u)]+\text{Sizesub}(T,u)+\sum_{u'\in S_r}[\text{Sizesub}(T,u)]$ $\leqslant 1+(2k+1)\text{Sizesub}(T,u)$. Part (b) follows. □

Now consider a forest $F_T^G$ formed by keeping each edge of a $r$-balanced tree $T$ with probability $1-1/G$. Let the color of any node in $F_T^G$ be same as its color in $T$.

**Lemma 4.** *Suppose $r>30$. Suppose $T$ is an $r$-balanced tree of size $l$, and $G\log^2 n \leqslant r \leqslant l \leqslant n$. Then, for some positive constant $c_1'$*

(a) *The probability that there exists a node $v$ in $T$, with $\text{Sizesub}(T,v)\geqslant r$, such that at most $1/(1.5G)$ fraction of $v$'s blue colored children in $T$ are not its children in $F_T^G$ is bounded by $l*n^{-c_1'\log n}$,*

(b) *With probability at least $1-l*n^{-c_1'\log n}$, for any node $v$ in $T$ with $\text{Sizesub}(T,v)\geqslant r$, $\sum_{\{u\in S_v\}}\text{Sizesub}(T,u)\geqslant \text{Sizesub}(T,v)/(4G)$, where $S_v=\{u \mid u$ is a blue colored child of $v$ in $T$ but is not a child of $v$ in $F_T^G\}$.*

(c) *With probability at least $1-l*n^{-c_1'\log n}$, the number of nodes, with no red ancestor, in the tree of $F_T^G$ containing the root is bounded by $l/r+2l\mathrm{e}^{-(\log l)/(8G\log r)}$.*

**Proof.** (a) Consider any node $v$ in $T$ such that $\text{Sizesub}(T,v)\geqslant r$. Suppose $v$ has $k$ blue children. Note that the total number of children of $v$ is at least $r/3$, at most two of which are red (by Lemma 3(a) and Proposition 2(a)). Thus, $k\geqslant r/3-2$. The probability that any blue child of $v$ in $T$ is not its child in $F_T^G$ is $1/G$. Thus, by Chernoff bounds, for some positive constant $c_1''$, the probability that at most $1/(1.5G)$ of $v$'s blue colored children are not its children in $F_T^G$ is bounded by $2^{-c_1''(r/3-2)/G}\leqslant 2^{-c_1'r/G}\leqslant 2^{-c_1'\log^2 n}=n^{-c_1'\log n}$, for some positive constant $c_1'$. Part (a) now follows since there are at most $l$ nodes $v$ in $T$ with $\text{Sizesub}(T,v)\geqslant r$.

(b) By part (a), with probability at least $1-l*n^{-c_1'\log n}$, for any node $v$ in $T$ with $\text{Sizesub}(T,v)\geqslant r$, at least $1/(1.5G)$ fraction of $v$'s blue colored children in $T$ are not its children in $F_T^G$.

Suppose the above holds. Consider a node $v$ with $k$ blue children in $T$, and satisfying $\text{Sizesub}(T,v)\geqslant r$. Let $S_v=\{u \mid u$ is blue colored child of $v$ in $T$ but is not a child of $v$ in $F_T^G\}$.

$$\begin{aligned}\sum_{\{u\in S_v\}}\text{Sizesub}(T,u) &\geqslant \sum_{\{u\in S_v\}}\frac{\text{Sizesub}(T,v)-1}{2k+1} \quad \text{(by Lemma 3(b))}\\ &\geqslant \frac{k}{1.5G}*\frac{\text{Sizesub}(T,v)-1}{2k+1}\\ &\geqslant \frac{\text{Sizesub}(T,v)}{4G}.\end{aligned}$$

(Note that $\text{Sizesub}(T,v)\geqslant r>30$ and $k\geqslant r/3-2\geqslant 8$.)

(c) Assume, without loss of generality, that $l\geqslant r^4$ (otherwise part (c) trivially holds, since $2l\mathrm{e}^{-(\log l)/(8G\log r)}$ would be $\geqslant l$). Note that any blue node, which has no red ancestor and is at level $\leqslant(\log l)/(2\log r)$, satisfies $\text{Sizesub}(T,u)\geqslant l(1/r)^{(\log l)/2(\log r)}=r^{(\log l)/(2\log r)}\geqslant r$.

Let the tree in $F_T^G$, which contains the root of $T$, be called $T_{\text{root}}$. Thus, (using part (b)) with probability at least $1 - l * n^{-c_1' \log n}$, we have that the number of blue nodes (in $T_{\text{root}}$) which have no red ancestor and are at level $>(\log l)/(2 \log r)$ in $T_{\text{root}}$ is bounded by

$$l\left(1 - \frac{1}{4G}\right)^{(\log l)/(2 \log r)} \leqslant l\left(1 - \frac{1}{4G}\right)^{(4G)(\log l)/(8G \log r)} \leqslant l e^{-(\log l)/(8G \log r)} \leqslant 2l e^{-(\log l)/(8G \log r)}.$$

On the other hand, since any node in $T$ has at most $r$ blue children, the number of nodes at level $\leqslant(\log l)/(2 \log r)$ in $T$, with no red ancestor, is bounded by $r^{1+(\log l)/(2 \log r)} \leqslant l/r$. Part (c) follows. □

**Corollary 5.** *Suppose $r > 30$. Suppose $T$ is an $r$-balanced tree of size $l$ and $G \log^2 n \leqslant r \leqslant l \leqslant n$. Let $B(l,r) = \min(l, (3l \log l)/r + 2l * e^{-(\log l)/(8G \log r)})$. Then, $P_T^G(B(l,r)) \leqslant n^{-c_1 \log n}$ for some positive constant $c_1$.*

**Proof.** Suppose $v$ is the root of $T$. Then by Lemma 4(c) and Proposition 2(f) we have that, for large enough $n$, with probability at least $1 - l * n^{-c_1' \log n} \geqslant 1 - n^{-c_1 \log n}$ for some positive constant $c_1$,

$$\text{Sizesub}(F_T^G, v) \leqslant \min\left(l, \frac{2l \log l}{r} + \frac{l}{r} + 2l * e^{-(\log l)/(8G \log r)}\right) \leqslant B(l,r)$$

Corollary follows. □

We will now show that for any binary tree $T$, one can transform it to a (possibly non-binary) $r$-balanced tree $T'$, such that $P_T^G(w) \leqslant P_{T'}^G(w)$, for all $w$. This would allow us to use the bound in Corollary 5 for all binary trees.

First, it is easy to see the following proposition.

**Proposition 6.** *For any tree $T_1$, if $v_1$ is a proper ancestor of $v_2$ in $T_1$ and $T_1''$ is obtained from $T_1$ by deleting the edge from* parent$(v_2)$ *to $v_2$ and adding $v_2$ as a child of $v_1$, then for all $w$, $P_{T_1}^G(w) \leqslant P_{T_1''}^G(w)$.*

Now, suppose $T_1$ is a tree such that subtree rooted at node $v$ of $T_1$ is binary. Then, the following procedure, Balance$(r, T_1, v)$ "balances" the node $v$ of tree $T_1$.

Balance$(r, T_1, v)$
(Here $v$ is a node of tree $T_1$.)
Let $s$ be the size of the subtree rooted at $v$.
Initialize $T_1^* = T_1$.
While there exists a proper descendant $u$ of $v$ such that
(i) $u$ is not a child of $v$, and
(ii) Sizesub$(T_1^*, u) \geqslant s/r$ and Sizesub$(T_1^*, u') < s/r$ for

each child $u'$ of $u$.
Pick one such $u$.
In tree $T_1^*$ delete the edge from parent($u$) to $u$ and add $u$ as child of $v$.
Endwhile
Return $T_1^*$.
End

The following proposition shows that the above procedure makes node $v$ "$r$-balanced".

**Proposition 7.** *Suppose $T_1$ is a tree, $v$ is a node in $T_1$ such that the subtree rooted at $v$ is binary. Suppose* Sizesub$(T_1, v) = s$. *Then, for the tree $T_1^*$ obtained by executing* Balance$(r, T_1, v)$, *the following hold*

(a) Sizesub$(T_1^*, u) \leqslant 2\lceil s/r \rceil - 1$, *for any child $u$ of $v$ (in $T_1^*$).*
(b) *For all but at most two children $u$ of $v$ (in $T_1^*$),* Sizesub$(T_1^*, u) \geqslant \lceil s/r \rceil$.
(c) *For each $w$, $P_{T_1}^G(w) \leqslant P_{T_1^*}^G(w)$.*
(d) *Subtrees rooted at each child of $v$ is binary.*

**Proof.** (a) Note that Sizesub$(T_1^*, u')$ is less than $s/r$ for any grandchild $u'$ of $v$ in $T_1^*$. Thus Sizesub$(T_1^*, u) \leqslant 1 + 2(\lceil s/r \rceil - 1)$, for any child $u$ of $v$ in $T_1^*$.

(b) For any new child $u$ of $v$ in $T_1^*$ (i.e. $u$ not a child of $v$ in $T_1$) Sizesub$(T_1^*, u) \geqslant s/r$. (b) follows since $v$ has at most 2 children in $T_1$.

(c) Follows using proposition 6.

(d) Follows from construction, since the procedure Balance$(r, T_1, v)$, does not increase the number of children of any proper descendant of $v$. □

Thus, the procedure Balance$(r, T_1, v)$ nearly balances the node $v$ of $T_1$. Transform a binary tree $T$ to $T'$ by repeatedly using Balance$(r, \cdot, \cdot)$, for balancing all the nodes of $T$, where a parent is balanced before its child (for example, by balancing in the dfs order of nodes in $T$). It is now easy to verify using Proposition 7 that $T'$ is $r$-balanced, and for each $w$, $P_T^G(w) \leqslant P_{T'}^G(w)$.

It immediately follows using Lemma 4 that

**Lemma 8.** *Suppose a binary tree $T$ is given with $l \leqslant n$ nodes. Suppose $r$ is given such that $G \log^2 n \leqslant r \leqslant l$. Let $B(l, r) = \min(l, (3l \log l)/r + 2l * e^{-(\log l)/(8G \log r)})$. Then $P_T^G(B(l, r)) \leqslant n^{-c_1 \log n}$ for some positive constant $c_1$, and large enough $n$.*

## 4. Branch and bound procedure

We now present our algorithm for branch and bound when communication costs are high. The following algorithm is a local best first algorithm with a bias towards children remaining in the processor where their parents are expanded. Initially, root

is present at processor 1. Each processor maintains two data structures: (1) a local priority queue of unexpanded nodes, and (2) a bound LCLD on the least cost leaf discovered. Each processor executes the following algorithm.

BandB1
Let LCLD $= \infty$.
**Repeat**
1. Node Expansion Phase
Repeatedly execute the following loop for a total time usage[2] of $G \log p$ steps (if one of the last $G$ steps involves transmission of a node, then complete that transmission before going to step 2).
**begin Loop**
1.1 Expand the least cost unexpanded node, $v$, in the local queue.
1.2 If $v$ is a leaf then let LCLD $=$ min(LCLD, Cost($v$)).
1.3 Otherwise, for each child $w$ of $v$:
1.3.1 With probability $1/G$, transmit $w$ to a random processor.
1.3.2 With probability $1 - 1/G$, add $w$ to the local queue.
1.4 Update the local queue with any arrivals from outside.
**end Loop**
2. Termination Detection Phase
Find the minimum, $m$, of LCLD on all the processors and update LCLD to $m$.
Halt if no processor has any unexpanded node with cost less than $m$.
**forever**
**end**

Note that the termination detection phase can be executed in $O(G \log p)$ steps. Thus, from now on we assume that the processors do not execute the Termination Detection Phase but automatically know when to stop. This would only affect the run time by a constant multiplicative factor and an additive factor of $O(G \log p)$. Note that no processor expands any node in $H - \tilde{H}$, if it has some node in $\tilde{H}$ left to expand. Thus, only the expansion of nodes in $\tilde{H}$ (and corresponding possible transmission of children of nodes in $\tilde{H}$) can affect the performance of the algorithm. Let $\hat{H}$ denote $\tilde{H}$ plus the children of the leaves of $\tilde{H}$. Based on the above discussion, for the analysis below we may assume without loss of generality that $H = \hat{H}$. Note that the size of $\hat{H}$ is at most 3 times the size of $\tilde{H}$, and height of $\hat{H}$ is at most one more than the height of $\tilde{H}$.

[2] Note that node expansion takes unit time step, node transmission takes $G$ time steps, and other operations do not take any time.

From now on, we let $n$ denote the number of nodes in $\hat{H}$ and $h$ denote the height of $\hat{H}$ (though earlier in the paper we based $n$ and $h$ on the size and height of $\tilde{H}$, the modification only effects the time bounds by a constant multiplicative factor).

Furthermore, for ease of giving the analysis we assume that the randomness in step 1.3 of BandB1, is shifted to the beginning of the algorithm. That is, before the start of the algorithm, for every node $v$ (except the root) in $H$, $v$ is ticked with probability $1/G$. In the step corresponding to step 1.3, a node $v$ is randomly transmitted to another processor, iff $v$ is ticked. This clearly does not change the probabilistic time bounds on the runtime of the algorithm above. The following describes the modified algorithm.

BandB2

Each node (except the root) in $H$ is ticked with probability $1/G$.

**Repeat**

Expand the least cost node, $v$, in the local queue.
If any of $v$'s children is ticked, then transmit it to a random processor. Unticked children are retained in the local queue.
Update the local queue with any arrivals.

**forever**

**end**

We dropped the reference to LCLD since it is used only for detection of termination. As discussed above, we assume that the algorithm automatically stops once the least cost leaf is discovered.

It is easy to see that the running time of the above procedure has same probabilistic time bounds (except for constant multiplicative factor, and additive factor of $O(G \log p)$) as the procedure BandB1. We prove the following theorem about the run time of BandB2 in the next section.

**Theorem 9.** *Suppose $\hat{H}$ has $n$ nodes, and height of $\hat{H}$ is $h$. Let $p$ be the number of processors (we assume $p < n^{1/12}$) and $G$ be the communication cost as described above. Suppose $r$ and $q$ are such that*
(a) $q < n^{1/12}$,
(b) $r > \max(Gp \log q, G \log^2 n)$, *and*
(c) $\log q > 8G \log r \log(Gp)$,
*then the algorithm BandB2 completes execution in time $O(hq^2 + n/p)$ with probability $\geqslant 1 - n^{-c_0 \log n}$, for some positive constant $c_0$.*

**Corollary 10.** *Suppose $p = 2^{\log^c n}$, where $c < 1/3$ and $G = b \log p$. Then the algorithm for branch and bound given above completes the execution in time $O(hq^2 + n/p)$ with probability $\geqslant 1 - n^{-c_0 \log n}$, for some positive constant $c_0$, where $q = O(n^{\varepsilon})$, for any constant $\varepsilon > 0$.*

**Proof.** Let $r = Gp\log^2 n$ and $q = 2^{8G\log r\log(Gp)} + 1$. Note that $q \leqslant 2^{8G\log(p^2\ \log^2 n)\log(Gp)} + 1 \leqslant 2^{c'\log^3 p} \leqslant n^{\varepsilon}$, for some positive constant $c'$, any $\varepsilon > 0$, and large enough $n$. Corollary now follows from Theorem 9. □

Thus for communication time logarithmic in the number of processors, branch and bound can be implemented in nearly optimal time, for $p < 2^{\log^c n}$, $c < 1/3$. (The "nearly" in the previous statement is due to the multiplicative factor of $q^2$ in $h$. As long as $h$ is smaller than $n/(pq^2)$, the algorithm is optimal. For example, if $h$ is not more that $n^{1-\varepsilon}$, for some positive constant $\varepsilon$, the algorithm is optimal.)

## 5. Analysis of the algorithm: Proof of Theorem 9

For a given $H$ ($\hat{H}$), where $\hat{H}$ has $n$ nodes and is of height $h$, fix $p$, $G$, $r$ and $q$ as in Theorem 9. In this section we will show the bound on execution time of BandB2 as claimed in the theorem.

Let $\hat{H}_0, \hat{H}_1, \hat{H}_2, \ldots$ denote a fixed disjoint partition of $\hat{H}$ into subtrees, such that the subtrees are of size between $q$ and $2q$, except possibly for one subtree which is of size $< q$. Note that the number of $\hat{H}_i$'s is at most $1 + n/q$.

Consider a forest $F$ formed from $\hat{H}$ by deleting the edge $(\text{parent}(v), v)$ for each ticked node $v$ in $\hat{H}$. For use in analysis, we place $G$ marks on $\text{parent}(v)$ for each such deleted edge $(\text{parent}(v), v)$. This would represent the extra work done when expanding $\text{parent}(v)$. A node with marks placed on it is sometimes refered to as marked node. Note that there is one-to-one correspondence between ticked nodes and the roots of trees in $F$. In the following lemma we claim that $F$ satisfies certain properties with high probability. $B(l, r)$ is as defined in Lemma 8.

**Lemma 11.** *For large enough $n$, with probability at least $1 - n^{-c_2\log n}$ (for some positive constant $c_2$), the following hold:*

(a) *For all $i$, the intersection of any tree in $F$ with $\hat{H}_i$ has at most $B(2q, r)$ nodes.*
(b) *For all $i$, the number of marks placed on nodes in $\hat{H}_i$ is at most $8q$.*
(c) *The sum of sizes of trees in $F$ which are not subtrees of some $\hat{H}_i$ is at most $2B(2q, r) * (1 + n/q)$.*

**Proof.** For a fixed $i$:

By Lemma 8, for any particular node $v$ in $\hat{H}_i$, probability that $\text{Sizesub}(\hat{H}_i \cap F, v) \geqslant B(2q, r)$, is at most $n^{-c_1\log n}$. Since there are a total of at most $n$ nodes in $\hat{H}_i$, the probability that for some $v$ in $\hat{H}_i$, $\text{Sizesub}(\hat{H}_i \cap F, v) \geqslant B(2q, r)$, is at most $n * n^{-c_1\log n}$.

The total number of children of nodes in $\hat{H}_i$ is at most $2 * 2q$. The probability that the number of marks in $\hat{H}_i > 8q$ is the probability that at least $8q/G$ children (in $\hat{H}$) of the nodes in $\hat{H}_i$ are ticked. Thus, by Chernoff bounds, the probability of this happening is at most $2^{-c_2' q/G} \leqslant 2^{-c_2'\log^2 n} = n^{-c_2'\log n}$, for some positive constant $c_2'$.

Thus, the probability that for some $i$, (a) or (b) fails to hold is bounded by $n*(n*n^{-c_1\log n}+n^{-c_2'\log n})\leqslant n^{-c_2\log n}$, for some positive constant $c_2$ and large enough $n$.

For (c) note that a tree in $F$ which is not a subtree of any $\hat{H}_i$ must contain an edge connecting two distinct $\hat{H}_i$'s in $\hat{H}$. Since there are at most $1+n/q$ edges connecting different $\hat{H}_i$s, whenever (a) holds, the sum of sizes of trees in $F$ which are not subtrees of some $\hat{H}_i$ is bounded by $2B(2q,r)*(1+n/q)$ (since each edge connecting two different $\hat{H}_i$'s could give rise to subtrees of size at most $B(2q,r)$ on the two $\hat{H}_i$'s it connects). Lemma follows. □

We say that $F$ formed from $\hat{H}$ is good iff it satisfies (a)–(c) in Lemma 11. Thus, $F$ is good with probability at least $1-n^{-c_2\log n}$, for large enough $n$.

For $S$, a tree in $F$, let weight of $S$ $=\mathrm{Size}(S)+$ the number of marks on nodes in $S$. Weight of $S$ thus represents the total work done by the processor expanding nodes in $S$ (and possibly transmitting their children).

If a tree $S$ in $F$ is a subtree of some $\hat{H}_j$, then we say that $S$ is of type $A$; otherwise we say that $S$ is of type $B$. Let $n_i$ denote the number of trees in $F$ of type $A$ and weight $i$.

**Proposition 12.** *Suppose $F$ is good. Then,*

(a) *Each tree of type A has weight at most* $10q$.

(b) *Total number of nodes in trees of type B is bounded by* $2(1+n/q)*B(2q,r)$. *Thus total weight of trees of type B is bounded by* $2(1+n/q)*B(2q,r)*(2G+1)$.

(c) $\sum_i[i*n_i]\leqslant(2q+8q)(1+n/q)\leqslant 20n$.

(a) and (c) in the above proposition follow from Lemma 11(b). (b) follows from Lemma 11(c).

**Lemma 13.** *Suppose $F$ is good. Fix a node $v$ of $\tilde{H}$. Then, the algorithm BandB2 expands $v$ within time $c_4(hq^2+n/p)$ with probability $\geqslant 1-n^{-c_3\log n}$, for some positive constants $c_3$ and $c_4$.*

**Proof.** We call ancestors of $v$ (in $\hat{H}$) special nodes and the trees in $F$ which contain any ancestor of $v$ as special trees. Clearly, at any particular time instant, at most one processor has a special node in its local queue. We refer to such a processor as the special processor (for that time instant).

Note that the total time taken before $v$ is expanded consists of

T1: time taken to expand $v$ and its ancestors and possibly transmit their children, and

T2: delay encountered by the ancestors of $v$ due to special processor expanding a non-special node (of lower cost) and subsequent possible transmission of its children.

Time delays in (T2) can be caused by one of the following:

T2.1: Special processor expands a node (or transmits a child of a marked node) from a tree of type $B$.

T2.2: Special processor expands a node (or transmits a child of a marked node) in a special tree (of Type $A$) of $F$.

T2.3: (post-delay) Special processor expands a node in a non-special tree $S$ of type $A$ (or transmits a child of a marked node in $S$), such that the root of $S$ arrived at the special processor after the root of the tree (in $F$) containing the special node.

T2.4: (pre-delay) Special processor expands a node in a non-special tree $S$ of type $A$ (or transmits a child of a marked node in $S$), such that the root of $S$ arrived at the special processor before the root of the tree containing the special node.

The above division of delays in groups is similar in spirit to the division in [4]. In the following subsections, assuming that $F$ is good, we will show that

(i) time taken due to T1 is bounded by $h*(2G+1)$;
(ii) time taken due to T2.1 is bounded by $2(n/q+1)*B(2q,r)*(2G+1)$;
(iii) time taken due to T2.2 is bounded by $h*B(2q,r)*(2G+1)$;
(iv) time taken due to T2.3 is bounded by $40n/p+100n^{1/4}q^2$, with probability at least $1-n^{c_5''\log n}$, for some positive constant $c_5''$.
(v) Time taken due to T2.4 is bounded by $a[(n^{1/4}+h)(100q^2)+20n/p]$, with probability at least $1-n^{-c_6\log n}$, for some positive constants $a$ and $c_6$.

Using the above bounds on T1, T2.1–T2.4, we have that:

With probability at least $1-n^{c_7\log n}$, for some positive constant $c_7$, the total time taken before $v$ is expanded is bounded by

$$
\begin{aligned}
&h*(2G+1)+2(2G+1)*(n/q+1)*B(2q,r)\\
&\quad+h*B(2q,r)*(2G+1)+\left[\frac{40n}{p}+100n^{1/4}q^2\right]\\
&\quad+a\left[(n^{1/4}+h)(100q^2)+\frac{20n}{p}\right],\\
&\quad\leqslant 3hG+6GnB(2q,r)/q+3hGB(2q,r)+40n/p+100n^{1/4}q^2\\
&\qquad+100an^{1/4}q^2+100ahq^2+20an/p\\
&\quad\leqslant \mathrm{O}((n/q)B(2q,r)G+hq^2+n^{1/4}q^2+n/p),\\
&\qquad\text{(Note that } G\leqslant q \text{ and } B(2q,r)\leqslant 2q)\\
&\quad=\mathrm{O}\left((nG/q)\left[\frac{2q\log q}{r}+2q\mathrm{e}^{-(\log q)/(8G\log r)}\right]+hq^2+n^{1/4}q^2+n/p\right),\\
&\quad=\mathrm{O}\left(\frac{nG\log q}{r}+nG\mathrm{e}^{-(\log q)/(8G\log r)}+hq^2+n^{1/4}q^2+n/p\right)\\
&\quad=\mathrm{O}(hq^2+n/p),\\
&\qquad(\text{since } q,p<n^{1/12}, r>Gp\log q, \text{ and } \log q>8G\log r\log(Gp)).
\end{aligned}
$$

This proves Lemma 13. We now proceed to show the above bounds on T1, T2.1–T2.4.

*Time taken due to* T1: Since there are at most $h$ special nodes, each having at most 2 children which may be transmitted, the time taken due to T1 is bounded by $h*(2G+1)$.

*Time taken due to* T2.1: Since there are at most $2(1+n/q)*B(2q,r))$ nodes in trees of type $B$, the total time taken due to T2.1 is bounded by $2(n/q+1)*B(2q,r)*(2G+1)$.

*Time taken due to* T2.2: Since there are at most $h$ special trees, and each tree of type $A$ has size at most $B(2q,r)$, time taken due to T2.2 is bounded by $h*B(2q,r)*(2G+1)$.

*Time taken due to* T2.3: For any tree of type $A$ to cause a post-delay, the root of the tree must be sent to the special processor. Probability of this happening is $1/p$.

If $n_i \geqslant n^{1/4}$, then, using Chernoff bounds, the probability that the total amount of post-delay due to nodes in trees of type $A$ and weight $i$ is greater than $(2n_i/p)i$, is bounded by $2^{-c_5' n_i/p} \leqslant 2^{-c_5' n^{1/4}/p}$, for some positive constant $c_5'$. On the other hand, if $n_i < n^{1/4}$, then the total amount of post-delays due to nodes from trees of type $A$ and weight $i$ is clearly $\leqslant n^{1/4}*i$.

Thus, if $p<n^{1/5}$, then with probability at least $1-n^{-c_5'' \log n}$ (for some positive constant $c_5''$), total amount of post-delay is bounded by $\sum_i [2*i*n_i/p] + \sum_i [n^{1/4}*i] \leqslant 40n/p + n^{1/4}*(10q)(10q+1)/2$ (since $\sum_i i*n_i \leqslant 20n$ and maximum value of $i$, the weight of a tree of type $A$, is bounded by $10q$).

Thus, time taken by T2.3 is bounded by $40n/p + 100n^{1/4}q^2$, with probability at least $1-n^{c_5'' \log n}$, for some positive constant $c_5''$.

*Time taken due to* T2.4: We now consider pre-delays. To model the pre-delays we consider the following queuing problem. In this problem the goal of the adversary is to get a high payoff with significant probability.

There are $k$ groups of customers, $X_1, X_2, \ldots, X_k$. Some groups of customers are of type $A$ and some are of type $B$. Number of customers in any group of type $A$ is bounded by $10q$. Number of groups of customers of type $A$ having $i$ customers is $n_i$. Total number of customers in groups of type $B$ is bounded by $2(2G+1)(1+n/q)B(2q,r)$. Also $\sum_i n_i * i \leqslant 20n$. Some groups contain special customers (such groups are called special). Total number of special customers in all the groups is bounded by $h$. $X_1$ is a special group.

At the start of the queuing process, group $X_1$ is assigned to processor 1. The queuing process alternates between sequences of arrival and service phases. At each service phase, one of the customers (if present) in each queue is serviced. In an arrival phase, a group of customers arrives. Each such group is assigned to a random processor (by assigning a group to a processor, we mean assigning all members of the group to the processor). A special group can arrive only if no special customers are present at any processor. The choice of arrival/service phase and the group at any arrival phase depends only on the the random choices (of processor assignment) made earlier in the queuing process. Note that at any step at most one processor has a special customer. Let the processor having a special customer be called special processor.

*Payoff*: The payoff (to adversary) for service phases is computed as follows. If the special processor (if any) in the service phase serves a customer from a group, $X_i$,

such that

$X_i$ is of type $A$,
$X_i$ is not a special group, and
$X_i$ arrived before the special customers,

then the payoff is 1 unit. Otherwise there is no payoff.

It is easy to see that the following holds.

**Proposition 14.** *There exists an adversarial strategy so that the probability of the adversary getting a payoff of* $\geqslant x$, *in the above queuing model*, *is at least as much as the probability of pre-delays* (*time taken due to T*2.4) *being* $\geqslant x$.

Further, we may assume, without loss of generality, that any group contains at most 1 special customer (we can consider the last special customer served in any special group to be the special customer of that group). Also, without loss of generality, we can assume that the special groups consist only of one customer which is a special customer.

The goal of the adversary is to maximize the chances that the payoff in the above process exceeds $a[(n^{1/4} + h)(100q^2) + 20n/p]$ (where $a$ is a constant given by Lemma 16 below).

A destination sequence, $d_1, d_2, \ldots, d_k$, is a sequence of numbers, each from the set $\{1, \ldots, p\}$. We interpret $d_i$ as the processor to which the $i$th arrival in the above process goes. Clearly, each sequence of random choice in the above process has an associated unique destination sequence.

The following proposition can be proved essentially along the lines of Proposition 7 in Chapter 5 of [7].

**Proposition 15.** *The following strategy maximizes the payoff for the adversary for any fixed destination sequence.*
1. *Schedule no arrivals while a special customer is present.*
2. *Always serve other customers before a special customer.*
3. *Schedule no service phases*, *if there is no special customer present.*

We omit the proof of above proposition which is essentially the same as given in [7].

We can thus assume without loss of generality that the adversary follows the strategy as in Proposition 15. Before proceeding with the proof, we first state a special case of the above queuing process which was analyzed by Karp and Zhang.

**Lemma 16** (Zhang [7]). *Suppose there are* $\leqslant h$ *groups containing one special customer each and j groups* (*of type A*) *containing one non-special customer each. Then*

*the probability that the payoff to the adversary is greater than* $a(j/p+h)$ *is bounded by* $2^{-j^{c'}}$, *for some positive constants* $a$ *and* $c'$, *if* $p<j^{1/3}$.

We assume without loss of generality that constant $a$ above is $>1$. We now modify the process in favour of adversary as follows.

1. If at the special processor a customer from a group of type $B$ or a special customer is being served, then no service takes place at the other processors.
2. If, at the special processor, a non-special customer from a group of type $A$ with $i$ customers, is being served then only a customer from a group of type $A$ with $i$ customers, if present, is served at any other processor.

**Lemma 17.** *Fix* $i$. *Suppose* $p<n^{1/12}$. *Then, for large enough* $n$, *probability that the adversary gets a payoff of at least* $a(\max(i*n^{1/4}, i(n_i/p+h)))$, (*where* $a$ *is the constant as in Lemma* 16) *due to service phases in which the special processor serves a customer from a non-special group of type* $A$ *with* $i$ *customers, is bounded by* $n^{-c_6\log n}$, *for some positive constant* $c_6$.

**Proof.** If $n_i<n^{1/4}$ then clearly the bound holds. So assume $n_i\geqslant n^{1/4}$.

To prove the bound in the lemma, we assume (as an advantage to the adversary) that the goal of the adversary is to maximize the probability that the payoff due to service phases in which the special processor serves a customer from a group of type $A$ with $i$ customers, is at least $a(\max i*n^{1/4}, i(n_i/p+h))$. As an added advantage to the adversary (just for proving the bound for this fixed $i$) we allow the adversary to choose the destination for every group except the special groups and the groups of type $A$ with $i$ customers. Clearly, in this situation, all groups of customers, except the special groups and the groups of type $A$ with $i$ customers, can be ignored.

We are thus left with the following groups of customers:

(a) $n_i$ groups of $i$ customers each.

(b) $\leqslant h$ special customers.

Now it follows from Lemma 16 that the payoff to the adversary due to service phases in which a special customer from a group of type $A$, with $i$ customers, is served is bounded by $a(\max i*n^{1/4}, i(n_i/p+h))$, with probability at least $1-2^{-n^{c'/4}}\geqslant 1-n^{-c_6\log n}$, for some positive constant $c_6$ and large enough $n$. □ (Lemma 17)

Thus, by Lemma 17, the probability that the adversary gets a payoff of more than $\sum_i[a\max i*n^{1/4}, i(n_i/p+h)])$ is bounded by $n*n^{-c_6\log n}$ for some positive constant $c_6$, for large enough $n$. Note that $\sum_i[a\max(i*n^{1/4}, i(n_i/p+h))])\leqslant a[(n^{1/4}+h)\sum_i i+\sum_i in_i/p]\leqslant a[(n^{1/4}+h)(10q)(10q+1)/2+20n/p]\leqslant a[(n^{1/4}+h)(100q^2)+20n/p]$.

Thus, by Proposition 14, time taken for T2.4 is bounded by $a[(n^{1/4}+h)(100q^2)+20n/p]$, with probability at least $1-n^{-c_6\log n}$, for some positive constant $c_6$.

This completes the proof of Lemma 13.

### 5.1. Proof of Theorem 9

From Lemmas 11 and 13 it follows that, the probability that some node of $\tilde{H}$ is not expanded within time $c_4(hq^2 + n/p)$ is bounded by $n * n^{-c_3 \log n} + n^{-c_2 \log n} \leqslant n^{-c_0 \log n}$, for some positive constant $c_0$ and large enough $n$. Theorem 9 follows.

## 6. Lower bound on the runtime of our branch and bound procedure

Consider a tree such that $\tilde{H}$ is a complete balanced binary tree. Then the expected number of nodes which are expanded by processor 1 (which starts with the root) is at least $n(1-1/G)^{\log n}$. Thus with significant probability ($>1/(Gn)$), processor 1 does at least $(n/2)(1-1/G)^{\log n}$ node expansions (since the maximum work for any processor is bounded by $nG$). Thus for $G = \log p$, the algorithm can be optimal (with probability $>1-1/(Gn)$) only for $p < 2^{O((\log n)^{1/2})}$.

## 7. Conclusion

In this paper we gave a simple parallel algorithm (which is a modified version of an algorithm given by Karp and Zhang) for Branch and Bound problems. We showed that this algorithm performs nearly optimally for a modest number of processors, in a model where communication costs are high. It is easy to formulate several variants of the algorithm by allowing the probability bias to be dependent on the size of the local queue. However all such variants, though seemingly better, are very hard to analyze.

Our analysis is built upon the methods developed by Karp and Zhang. We could not use the simplification of Ranade, since Ranade's analysis crucially depended on symmetry of distribution of different nodes to different processors. It will be interesting to see if the techniques used here can be combined with Ranade's methods to simplify the analysis.

## Acknowledgements

I am specially grateful to Raghu Raghavan for introducing the problem to me and for constantly urging me to solve the problem. I would like to thank Mohan Kankanhalli, Timothy Poston, Raghu Raghavan and Weiguo Wang for several discussions we had on different ways of analyze the algorithm given in this paper and its variants. An anonymous referee suggested several changes which improved the presentation of the paper.

## References

[1] A. Bar-Noy, S. Kipnis, Designing broadcasting algorithms in the postal model for message-passing systems, In Proc. 4th Annual ACM Symp. on Parallel Algorithms and Architectures, ACM Press, 1992, pp. 11–22.

[2] D. Culler, R. Karp, D. Patterson, E. Santos, A. Sahay, K. Schauser, R. Subramonian, T. von Eicken, Towards a realistic model of parallel computation. In Proc. Principles and Practices of Parallel Programming, 1993.
[3] R. Karp, A. Sahay, E. Santos, K. Schauser, Optimal broadcast and summation in the logp model. In Proc. 5th Symp. on Parallel Algorithms and Architectures, July 1993.
[4] R. Karp, Y. Zhang, A randomized parallel branch and bound procedure. In Proc. ACM Annual Symp. on Theory of Computing, ACM Press, 1988, pp. 290–300.
[5] R. Karp, Y. Zhang, Randomized parallel algorithms for backtrack search and branch and bound computation, J. ACM 40(3) (1993) 765–789.
[6] A. Ranade, A simpler analysis of the Karp–Zhang parallel branch-and-bound method, Technical Report UCB/CSD 90/586, University of California, Berkeley, 1990.
[7] Y. Zhang, Parallel algorithms for combinatorial search problems; Ph.D. Thesis, University of California, Berkeley, 1989.